MultiMID - Multisession MIDI Overlay Driver
Version 1.08

Copyright © H. Seib, 1995-96

Hermann Seib
Am Tabor 12/1/24
A-1020 Vienna
Austria / Europe

Internet:
http://www.t0.or.at/~seib/
e-mail:seib@t0.or.at
or: seib@cybercafe.co.at

User's Manual

Contents

Introduction

This document describes **MultiMID**, the Multisession MIDI Overlay Driver.

MultiMID has been created to circumvent a glaring limitation of most currently available MIDI device drivers - they accept only ***one*** client program, session or connection. While this is nice as long as there is no more than one program active that uses the MIDI device, it becomes a real nuisance when there are more programs running concurrently that want to share the same device. Using Windows' standard MIDI driver, MIDI Mapper, it is not possible to open multiple MIDI sequencers without getting a Windows error message. Creative Lab's SoundBlaster MIDI drivers also only allow one client program to use its MIDI port at a time. The only way to open more than one MIDI program in Windows is to install a "multi-client" driver for your MIDI device. **MultiMID** is your solution!

If for example, you are a Waveblaster owner, you cannot run WBPanel (or my *much* better program to manipulate the Waveblaster, **WBMan**) and Cakewalk Apprentice concurrently - each of them wants to open the SB16 MIDI In and SB16 MIDI Out port, but the Soundblaster MIDI drivers are single-session drivers...

Enter MultiMID.

MultiMID can add an overlay or layer on top of other MIDI drivers. MultiMID makes it possible to have up to 8 simultaneous connections to an underlying MIDI driver - that should be enough for all musical purposes...

While the layering that MultiMID performs slows MIDI communications down a little bit (as MIDI messages have to be routed though, to or from the REAL MIDI driver), it is a very flexible approach in that it is not limited to any underlying hardware architecture. The communications overhead is neglectible considering the REAL MIDI data transfer rate of 31.25 kbaud.

There are two versions of MultiMID: a **demo version** and a **registered version**. The demo version is freely available for evaluation purposes. This documentation covers both versions, as the demo version is a strict subset of the registered version. This allows users of the demo version to decide whether they could benefit of the registered version. For details on how to obtain the registered version, please see the chapter **License/Warranty Disclaimer**.

Version Information

1.08	1996-05-28	loads other multimedia drivers directly instead of relying on Windows' multimedia functions
1.07	1996-03-29	additional safety measures against invalid pointers added
1.06	1996-03-11	Win95 compatibility enhanced; message stack size tripled
1.05	1996-01-31	segment alignment set to 2 by Watcom Linker; now set to 16; trace window added in debug version as writes to file don't work under Win95; plug-in names truncated in configuration; corrected; global plug-ins didn't get parameterized correctly
1.04	1995-12-11	cures a bug in V1.03 that sometimes inhibited MIDI output.
1.03	1995-11-16	first commercial version; adds plug-in modules and full configurability.
1.02	1995-09-21	adds the possibility to rename the generated devices;

		allows dummy MIDI IN devices (did not work correctly in V1.01)
1.01		1995-09-06 all device capabilities are now supported, including volume change and caching; correct device capabilites are reported for the generated devices; generated device names are beautified (i.e., cut at the last possible blank if the name is too long, instead of being cut at the last possible position);
1.00	1995-08-31	Start version

Installation

Installation is an easy process.

Unpack all the files that came with MultiMID into a temporary directory. The directory should contain the following files:

MULTIMID.DRV	the MIDI driver
OEMSETUP.INF	Control Panel installation information for the MIDI driver
MULTIMID.WRI	this documentation
HISTORY	detailed history of MultiMID
READ.ME	last minute changes
PLUGIN.H	header file for plug-in module creation (*not in demo version!)*

Then, install MultiMID using the Windows Control Panel. Open Drivers/Add/Unlisted, and enter your temporary directory. Control Panel should then display

MultiMID Multisession Overlay Driver

Select this and press **OK**.
That's all it takes to install the MultiMID driver.

After having installed the driver, the Control Panel opens MultiMid's configuration page for driver configuration (see **Configuration** below on how to configure MultiMid).

When Windows is started the next time, MultiMID is ready for operation!

Upgrading from Previous Versions

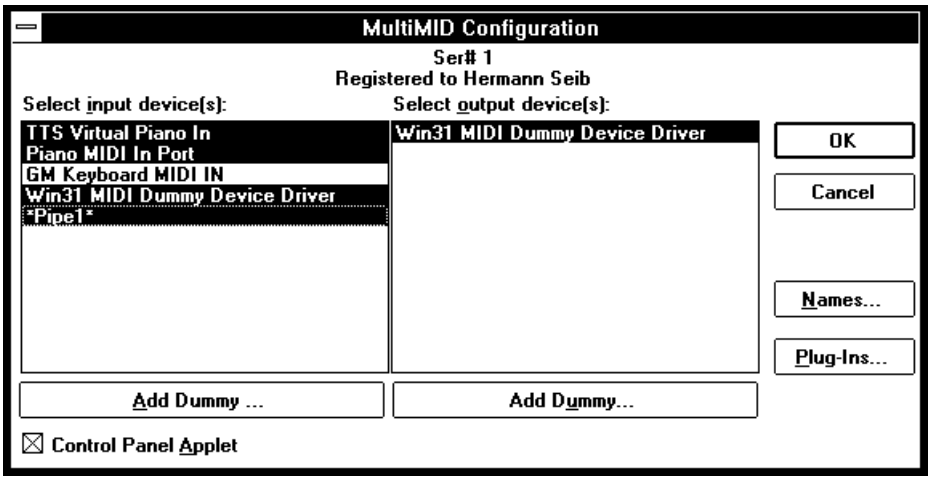
If you are upgrading from a previous version of MultiMID (**V1.00** to **V1.02**), the above procedure does ***not work***! The previous versions of MultiMID contained an invalid version number, which has been corrected in V1.03 - but it obviously causes serious malfunctions in the Control Panel. If you try to install the new version, Control Panel copies **the old version over the new version** instead - and continues to use the old version!

To upgrade in this special case, first **remove** the old driver with the Control Panel, then restart Windows, and then use File Manager or something comparable to **physically delete** the old driver from the WINDOWS\SYSTEM directory!

After that, the normal installation operation as described above can be applied.

Configuration

Upon installation, and if you select "Configure..." from the Control Panel's **Driver** section, you see the following dialog (it's contents will vary according to your MIDI setup):



Use this dialog box to select the input and output driver(s) to be overlayed.

MultiMID can overlay up to 8 input and 8 output devices. If more devices are selected, only the first 8 can be used.

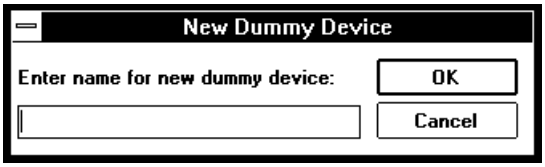
For each input device, i.e., **SB16 MIDI In**, MultMID will generate a dummy device named **Multi-SB16 MIDI In**. This driver will not be seen in MultiMID's configuration dialog. You will see it when you open a program client and access its MIDI devices input selection dialog.

For each output device, i.e., **SB16 MIDI Out**, MultMID will generate two dummy devices named **Multi-SB16 MIDI In** and **Pipe-SB16 MIDI Out**. These drivers will not be seen in MultiMID's configuration dialog. You will see it when you open a program client and access its MIDI devices output selection dialog.

If no Input and/or Output devices are available, a ***none*** selection will appear in MultiMID's configuration dialog box. When ***none*** is selected, MultiMID generates a dummy device called **Multi-*none***. This driver will not be seen in MultiMID's configuration dialog. You will see it when you open a program client and access its MIDI devices selection dialog. This device can be opened like any other MIDI device. This allows to run programs that *need* a MIDI input and/or Output device on machines that don't provide the necessary hardware and/or software. A **Multi-*none*** input device can be opened, but will never send MIDI data to the calling program (unless the data is sent through the accompanying **Pipe-*none*** output device); MIDI messages sent to a **Multi-*none*** output device are silently thrown away.

Additional Dummy Devices

The **Add Dummy...** buttons allow the addition of more dummy devices. If you press one of them, the following dialog appears:



Here, you can enter a new dummy device name. If you press OK, this device is added to the list in the main configuration dialog.

This button does not work in the demo version. For the demo version, you can use the following approach:

If you want to add more dummy devices, this is possible by directly editing the SYSTEM.INI file. Open SYSTEM.INI with your favourite ASCII editor and search for the [multimid.drv] section.
If you want to add a new pipe, add the line

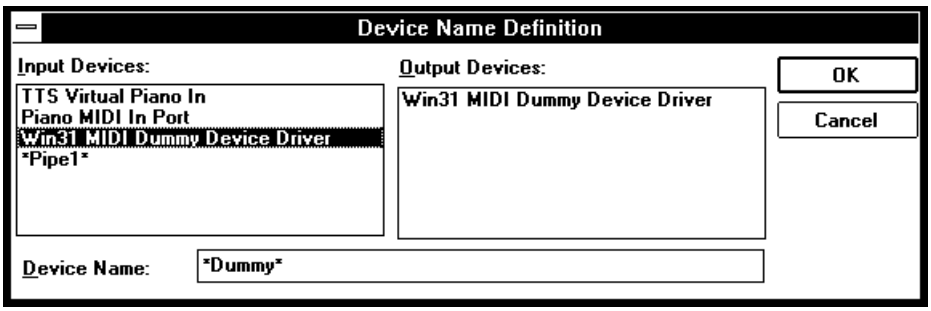
```
MidiIn2=Pipe1
```

*(provided you have already allocated one real MultiMID device). This will add the input device **Multi-Pipe1** and the output device **Pipe-Pipe1** to your system. After having finished editing, restart Windows and the new device drivers will be available.*

Attention: *If you reconfigure MultiMID by using its configuration dialog afterwards, the additional dummy devices added by manually editing the SYSTEM.INI file will be silently lost or thrown away.*

Device Names

The **Names...** button allows to rename the generated devices. If you press it, the following dialog appears:



Here, all devices that are highlighted in the main configuration dialog's list boxes are displayed. If you select one of the devices, you can add an alias name in the edit field at the bottom of the dialog.

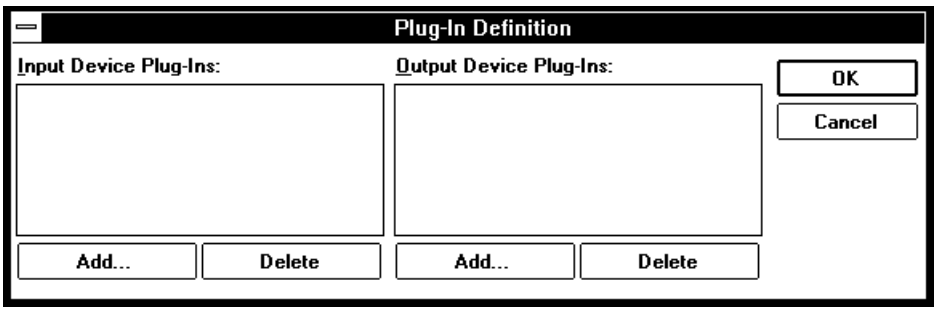
This has been implemented to allow the generated device names to be more concise. If, for example, you want to overlay a device called **This is a very long IN device**, the generated name would become **Multi-This is a very long IN**, which can be quite unreadable. Renaming it to **MyInput**, for example, would cause MultiMID to generate the devices **Pipe-MyInput** and **Multi-MyInput**.

This button does not work in the demo version.

Global Plug-In Modules

The **Plug-Ins...** button allows the definition of global plug-in modules.

When pressed, the followig dialog appears:

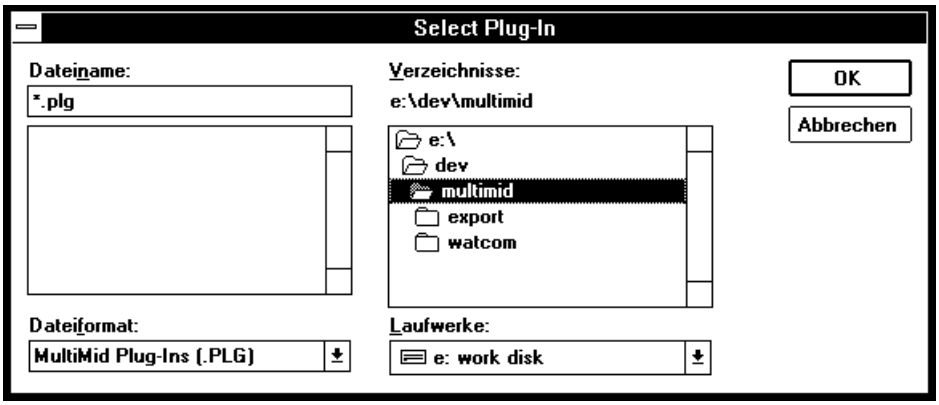


Here, you can configure MultiMID's global plug-in modules (see the **Plug-Ins** chapter for more information).

The left side handles plug-ins that handle incoming MIDI message, the right side handles outgoing MIDI messages.

To add a plug-in module, press one of the **Add...** buttons.
To delete a plug-in module, select the plug-in module and press the corresponding **Delete** button.

If you press one of the **Add...** buttons, the following dialog appears:



Here, you can select a global plug-in module. The dialog is a standard Windows 3.1 File Open dialog; the texts vary according to your Windows language/version configuration.

This button does not work in the demo version.

Control Panel Applet

Normally, MultiMID is configured by selecting **Configure...** in the **Drivers** applet of the Control Panel. The **Control Panel Applet** check box on the bottom of the main configuration dialog allows to define whether MultiMID appears as a separate icon in the Control Panel. When selected, the next time you activate the control panel it will show a new item:

This item, when selected, opens the normal MultiMID configuration dialog.

Operation

Once installed and configured, MultiMID provides a set of new MIDI input and output devices that can be selected from your MIDI programs.

If, for example, you configure MultiMID to overlay the following MIDI drivers:
Input: **SB16 MIDI In**
Output: **SB16 MIDI Out**,

MultiMID generates 3 new devices called
Input: **Multi-SB16 MIDI In**
Output: **Pipe-SB16 MIDI In**
Output: **Multi-SB16 MIDI Out**.

The **Multi-...** drivers overlay the MIDI Input and Output device drivers.
The **Pipe-...** driver allows client programs to send their output data to another client program's MIDI IN connection.

If you use the driver **Multi-SB16 Out** instead of the original **SB16 MIDI Out** Soundblaster driver in your MIDI sequencer program, you will be able to connect up to 7 other programs to the Soundblaster MIDI port simultaneously, (total of 8 programs).

Note: MultiMID allows up to 8 input and 8 output connections at a time; these 8 connections are used for *all* configured input or output devices. So, if you want, for example, to overlay 2 MIDI Input devices, and 6 programs have the first MultiMID Input device open, then only 2 programs can open the second MultiMID device. To open a third connection to the second MultiMID device, you have to close one of the other connections first.

MultiMID can be used with the MIDI Mapper, too; just create a new map with the selected **Multi-***whatever* output device and activate it.

Plug-In Modules

Plug-In modules are not available in the demo version.

MultiMID provides an interface for **plug-in modules**; up to 8 input and 8 output modules can be "plugged" into MultiMID's internal MIDI message processing system. These modules can be used to re-route and alter the processed MIDI messages before they are really sent to the target program or MIDI device.

Internally, a plug-in module is a special kind of callback function with the following prototype:

Syntax

DWORD PASCAL FAR **PlugInProc**(*wMsg*, *wDirection, dwParam*)

Parameters

WORD *wMsg*

specifies the message sent to the plug-in function; see below.

WORD *wDirection*

specifies the plug-in direction; possible values are:

0 ... input plug-in module invoked
1 ... output plug-in module invoked

DWORD *dwParam*

contains message-specific parameters; see below.

Return value

Unless specified otherwise under a specific message, the plug-in function should return MMSYSERR_NONE (or **0**, which is equivalent).

Adding Plug-In Modules

There are two methods to add a plug-in module to MultiMID: *load-time* and *run-time*.

Load-Time Adding of Plug-In Modules

Plug-In modules that are to be loaded upon MultiMID's startup have to reside in a DLL. The Plug-In selection dialogs in MultiMID's configuration suggest a file extension of **.PLG** for the plug-in modules, but this is neither mandatory nor necessary; it just helps to distinguish them from other DLLs.

A plug-in module located in a DLL must export the plug-in as a function called **PlugInProc**. This function is then used by MultiMID. There can be only one plug-in module in the DLL.

Plug-in modules added at load time are always global (see below). They are not removed until MultiMID is removed from memory.

Run-Time Adding of Plug-In Modules

This can be done with two special messages (see below for message specification); one for loading the Plug-In module, and one for removing it. Plug-in modules added at run time are removed by MultiMID when the module that added them closes the connection with MultiMID.

Plug-in modules added at run time can be either global or local (see below).

Plug-In Types

There are two types of plug-in modules: *local* and *global*.

Local Plug-In Modules

Local plug-in modules can be added at run time only. When a program adds a local plug-in module to MultiMID by sending a special message (see below), it is added to the overlayed device the program is connected to.

Global Plug-In Modules

Global Plug-Ins act on every MultiMID device.

Every type of plug-in can be added to input and output devices.

Plug-In Messages

PIM_INITIALIZE

This message is sent to the plug-in module when it is loaded.

dwParam is undefined.

The return value is **MMSYSERR_NOERROR** (0) if the plug-in function supports the passed *wDirection*, or **MMSYSERR_NOTSUPPORTED** if not.

PIM_TERMINATE

The plug-in function receives this code just before it is removed. This allows it to tidy up things.

dwParam is undefined.

The return value is undefined.

PIM_DEVICES

This call is used by MultiMID to parameterize the plug-in module. Sometimes, plug-in modules must know the attached devices; this message provides the necessary information.

dwParam is a **LPMMID_DEVINFO** pointer that points to the first entry of an array of **MMID_DEVINFO** structures. For an input plug-in, the array holds up to **MAX_SESSIONS** (8 in current version) devices; for an output plug-in, it holds up to **2*MAX_SESSIONS** devices (output/pipe-devices).

The format of a **MMID_DEVINFO** structure is:

```
typedef struct _MMID_DEVINFO            /* MultiMID device information       */
 {
 WORD   wAllocated;                    /* # allocated sessions              */
 HANDLE handle;                        /* overlayed device handle           */
 short  sReenter;                      /* reentrancy flag                   */
 PLUGIN plugin[MAX_PLUGINS];           /* plug-in definitions               */
 union
   {
   struct
     {
```

```
    MIDIINCAPS caps;                /* underlying device capabilities    */
    UINT   Started;                 /* flag whether int'l midiIn started */
    DWORD  StartTime;               /* internal midiInStart() time       */
    HWND   hWndAuto;                /* Window to be activated on Start   */
    } In;
  struct
    {
    MIDIOUTCAPS caps;               /* underlying device capabilities    */
    } Out;
  };
} MMID_DEVINFO, * PMMID_DEVINFO, FAR * LPMMID_DEVINFO;
```

For plug-ins, the fields *In.caps*, or *Out.caps*, respectively, are the most interesting items here. They contain the device capabilities of the overlayed devices.

A plug-in module may inspect the values of the **MMID_DEVINFO** structures freely; however, it ***must not*** change any of the settings, as the pointer passed in *dwParam* points directly into MultiMID's data areas.

Note: the values in the **MMID_DEVINFO** structures are not guaranteed to be valid; MultiMID initializes these when it receives the first **midiInGetDevCaps()** or **midiOutGetDevCaps()** call, which can be long after a plug-in module has been loaded. A plug-in module should assume the structures to be uninitialized as long as `Dev[0].In(Out).caps.szPname[0]` contains a null character.

The return value is undefined.

PIM_CLIENTS

MultiMID informs the plug-in module about client configuration changes. Every time a program opens or closes a MultiMID device, the plug-in modules for this device receive a notice.

dwParam is a **LPMMID_CLIENT** pointer that points to the first entry of an array of **MMID_CLIENT** structures. The array holds up to **MAX_SESSIONS** (8 in current version) clients. Unused clients have all entries set to NULL.

The format of a **MMID_CLIENT** structure is:

```
typedef struct _MMID_CLIENT             /* MultiMID client information       */
 {
 HANDLE hMidi;                          /* parent handle in MMSYSTEM         */
 DWORD  dwCallback;                     /* callback for client               */
 DWORD  dwInstance;                     /* reference data from client        */
 DWORD  dwFlags;                        /* allocation flags                  */
 WORD   wDevID;                         /* device ID                         */
 DWORD  startoff;                       /* midiInStart offset from internal  */
 LPMIDIHDR mhdr;                        /* SysEx input buffers               */
 } MMID_CLIENT, * PMMID_CLIENT, FAR * LPMMID_CLIENT;
```

A plug-in module may inspect the values of the **MMID_CLIENT** structures freely; however, it ***must not*** change any of the settings, as the pointer passed in *dwParam* points directly into MultiMID's data areas.

The return value is undefined.

PIM_CALLBACK

Ths message provides a call-back function address to the plug-in device. The plug-in module can use this entry point to "poke" messages into MultiMID without having to open a separate

connection.

dwParam is a pointer to the first of 2 **PLGCALLBACK** (see PLUGIN.H for the definition) pointers to MultiMID's **midMessage** and **modMessage** functions. These pointers are guaranteed to stay valid during the lifetime of the plug-in module. For the syntax and parameters of these functions and the messages that can be sent to them, please see the Microsoft Windows 3.1 Device Driver Kit (DDK).

A small example on how to convert the parameter into a function pointer and use it:

```
PLGCALLBACK midMessage = ((PLGCALLBACK)dwParam)[0];
PLGCALLBACK modMessage = ((PLGCALLBACK)dwParam)[1];
...
// simulate a MIDI message from a device!
rc = (*midMessage)(wDeviceID, MDM_POKEDATA, -1, 0x90407FL, 0);
```

The return value is undefined.

PIM_MIDI

This call is used for MIDI data processing.

dwParam is a LPMIDIHDR for the MIDI message to be processed. The following fields are guaranteed to be filled:

lpData	pointer to the data area
dwBufferLength	length of the allocated buffer
dwBytesRecorded	# bytes in buffer (for Midi IN Plug-Ins)
lpNext	pointer to additional messages (can be generated by previous plug-in); NULL if none available
dwUser	the connection for final output of the message
reserved	the device ID for final output of the message

The function returns a **LPMIDIHDR**. This can be:

NULL	suppress the message (i.e., filter it)
dwParam	the same pointer that came in (i.e., keep it); buffer contents can be changed, if needed
new LPMIDIHDR	contents of new **MIDIHDR** are used in further processing

If the callback function wants to return more than one MIDI message in response to the input, it can do so by putting the address of the second LPMIDIHDR into mhdr->lpNext.

Short MIDI messages are converted to long MIDI messages before walking the plug-in chain, and eventually converted back after processing has been completed.

If the plug-in module supplies another LPMIDIHDR instead of the one passed in *dwParam*, it is the responsibility of the module to do the housekeeping (i.e., proper GlobalAlloc() etc. of the header, GlobalFree() upon termination, etc.). Please note that input plug-in modules are called at **interrupt time**; the usual restrictions apply (see MIDI callback functions for details).

Attention Programmers: Special Messages

In addition to MultiMID's **Pipe-...** drivers, MultiMID provides a special feature that allows programs to send MIDI data to other program's input devices.

Some of the messages can be sent to MIDI Input devices only, others can be sent to Input and Output devices; this is indicated by the function called in the examples below.

The ***MDM_...*** definitions below are `#define`'d in PLUGIN.H (*and thus available to owners of the registered version only*).

Note: the message numbers have been changed from previous versions as Windows 95 didn't accept the old numbers; for compatibility, the old ***MDM_...*** definitions still work with MultiMID, but only with Windows 3.1 or 3.11. If you are using Windows 95, they cease to work (i.e., Win95 throws them back with an **MMSYSERR_INVALPARAM** code).

Short MIDI Messages

Programs can send MIDI data to the MIDI Input device by issuing

midiInMessage(hMidi, ***MDM_POKEDATA*** *(or* ***0x4001****)*, msg, ***0***);

where **hMidi** is the handle to the Input device and **msg** is the short MIDI message as a DWORD. The short MIDI message is then sent to all other programs that are connected to this MIDI Input device.

Long MIDI Messages

Programs can send SysEx information to the MIDI Input Device by issuing

midiInMessage(hMidi, ***MDM_POKELONGDATA*** *(or* ***0x4002****)*,
(DWORD) lpMidiOutHdr, ***0***);

where **msg** is a far pointer to a pre-filled MIDIHDR structure like the ones used for **midiOutLongMsg**(). The fields ***lpData*** and ***dwBufferLength*** must be filled; all others can be ignored. The SysEx MIDI message is then sent to all other programs that are connected to this MIDI Input device.

Auto Focus

A program can request to be activated automatically when another program starts MIDI Input on the same device.

midiInMessage(hMidi, ***MDM_AUTOHWND*** *(or* ***0x4003****)*, (HWND)hWndAuto, 0);

Whenever another program starts recording from this MIDI input device, the passed window is activated automatically.

Add Plug-In Module

This message is used to add a plug-in module at run time.

midi(In/Out)Message(hMidi, ***MDM_ADDPLUGIN*** *(or* ***0x4004****)*,
MAKELONG(type, pos),
(DWORD)(LPPLUGINFUNC)function);

type plug-in type:
0 ..... global plug-in module
!= 0 ... add to device identified by hMidi
pos position in chain; values:
0..MAX_PLUGINS-1 = defined position,
>= MAX_PLUGINS = last position
function pointer to a plug-in callback function

MultiMID return values :

0	OK
MMSYSERR_NOMEM	no slot free for plug-in module

*This message always returns **MMSYSERR_NOMEM** in the demo version.*

Delete Plug-In Module

This message is used to delete a plug-in module at run time.

midi(In/Out)Message(hMidi, ***MDM_DELPLUGIN*** *(or **0x4005**)*,
MAKELONG(type, 0),
(DWORD)(LPPLUGINFUNC)function);

type plug-in type; see above
function pointer to the callback function
if NULL, all callback functions for the connection are deleted

MultiMID return values :

0	OK
MMSYSERR_INVALPARAM	callback function not there

*This message always returns **MMSYSERR_INVALPARAM** in the demo version.*

Configure Devices

This message is used to make sure the device information in MultiMID is correct. This may be needed by external configuration programs, as MultiMID normally initializes its device information blocks the first time **midiInGetDevCaps()** or **midiOutGetDevCaps()** get called for one of MultiMID's devices, and *that* might occur *after* a plug-in configuration program has been started.

midi(In/Out)Message(hMidi, ***MDM_DEVCFG*** *(or **0x4006**)*, 0, 0);

MultiMID return values :

0	OK
MMSYSERR_INVALPARAM	callback function not there

*This message always returns **MMSYSERR_INVALPARAM** in the demo version.*

Frequently Asked Questions

OK, as this is the first version of MultiMID, I asked them myself; keep 'em rolling in, and I'll put them here in future versions!

Q: I have a little MIDI-THRU application that allows me to send data from the input to the output device; I used **Multi-XXX IN** as my input device and **Pipe-XXX IN** as my output device; now every time a note arrives on my XXX input port, the system comes to a screeching halt. What happens here?

A: You produced a nice little feedback loop. Every time you send a MIDI note to the **Pipe-XXX IN** output device, it's echoed on your **Multi-XXX In** device, this gets output to the **Pipe-XXX IN** device, and so on.

Q: Should I overlay *every* MIDI In / Out device with MultiMID?

A: No. Some devices already allow for multiple connections. If you don't need the **Pipe-...** feature of MultiMID, there is no sense in overlaying these. Unfortunately, the documentation for most MIDI device drivers is a bit sparse on that point; you have to try whether you can connect multiple programs to the same device. If the 2nd open fails, the driver is a candidate for MultiMID.

Q: I would like to install MultiMID under Windows 95. How can I do that?

A: First of all, MultiMID in its current version has been designed for Windows 3.1 (or 3.11, or WfW) only. It is an old multimedia driver and as such not the optimal choice for Windows 95. While it works under some circumstances, I've watched it killing Windows 95 quite often. You *can* install and use MultiMID under Windows 95 (but don't expect it to work well):

- Open the Control Panel (under the **Settings** item of the **Start** menu.
- Click on the **Add New Hardware** icon and follow the **Wizard** prompts.
- When asked 'Do you want Windows to search for your new hardware?', select **No**.
- When prompted to select the type of hardware you want to install.....select **Sound,Video and Game Controllers**.
- On the next dialog click on the **Have Disk** button.
- Use the **Browse** button to direct windows to the OEMSETUP.INF file in MultiMID's temporary directory.
- Select **OK** and and a dialog will appear highlighting 'MultiMID Multisession Overlay Driver'
- Select **OK** and **FINISH**. The required files will be copied from the temporary directory to your Windows directory.
- On completion the MultiMID configuration dialog will appear, from which you may set up your midi ports as per Windows 3.1x
- After configuring MultiMID, you will be asked to restart Windows for the changes to take effect.
- After re-starting Windows you may delete the contents of the temporary directory.

Q: What is MIDI?

A: An acronym... if you don't know it, you don't need MultiMID!

Order Form

If you want to order the registered version of MultiMID, please fill out the following form and send it to:

Hermann Seib
Am Tabor 12/1/24
A-1020 Vienna
Austria / EUROPE

Name: _____

Address: _____

_____

City: _____ State: _____ Zip: _____

MultiMID V1.08: Number of Copies _____ ($39.95 each): $_____

Upgrade to V1.08: Number of Copies _____ ($9.95 each): $_____

Serial# for upgrade: _____

Shipping and Handling: $ _____5

Disk Format: 5.25" [ ]
3.5" [ ]

Total Enclosed: $_____

Note: if you send an Eurocheck, you can substitute ATS 250.-- instead of the $39.95 above, and ATS 50.-- for S+H. So, when registering one copy of MultiMID, send me an Eurocheck and put "ATS 300.--" on it - Austrian banks don't charge the enormous $12-$15 overhead for Eurochecks.

License / Warranty Disclaimer

As stated above, MultiMID is a commercial program now; the added features, in my opinion, rectify that. However, there are two versions of MultiMID: a **demo version** and a **registered version**. The demo version is still in the public domain; no "30-day-trial period" or such stuff, just use it and spread it all over the world! If you like it, or have suggestions for further enhancements, send me a letter or a message to one of my e-mail addresses (see top page). "seib@t0.or.at" is preferred, as I check that nearly every day.

You may freely distribute the demo version of MultiMID V1.08 provided that no fee is charged for copying, distribution, or use, and that it is unmodified and distributed with all of its original accompanying files and documentation. If you want to include MultiMID in your products, please contact Hermann Seib for licensing plans.

Hermann Seib disclaims all warranties, express or implied, including but not limited to warranty of merchantability or fitness for a particular purpose, and will not be liable for any damages resulting from the use of this software, including loss of data. Use this software at your own risk.

MultiMID is copyrighted © 1995-96 by Hermann Seib.